VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Protokol IFTTT

IFTTT Protocol

2018 David Němec

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **David Němec**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Protokol IFTTT
IFTTT Protocol

Jazyk vypracování: čeština

Zásady pro vypracování:

Práce zabývající se protokolem pro vzájmenou interakci služeb a zařízení. Cílem je analyzovat tento protokol a jeho možnosti a navrhnout ukázkové řešení.

1. Prostudujte možnosti a vlastnosti protokolu IFTTT.
2. Detailně popište jeho možnosti a způsob použití z pohledu moderních webových aplikací.
3. Navrhněte způsob využití tohoto protokolu pro využití ve vazbě na existující webové systémy.
4. Implementujte knihovnu (C#, JS) pro integraci knihovny do stávajících systémů.
5. Navrhněte a realizujte konkrétní případové studie využití protokolu ve vazbě na reálný webový systém.
6. Zhodnoťte využití tohoto protokolu v praxi.

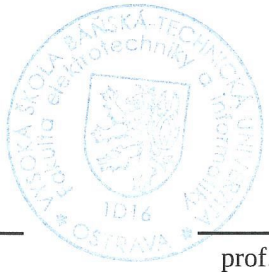
Seznam doporučené odborné literatury:

[1] Erixc Elliot: Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries, O'Reilly Media, 2014, ISBN: 978-1491950296
[2] Albert MArtinez: The Ultimate IFTTT Guide: Use The Web's Most Powerful Tool Like A Pro, Amazon ebook
[3] Matt Cole: Utilizing IFTTT: Second Edition: If I Read This Book Then I Will Learn Something, Amazon ebook
[4] Valerio de Sanctis: ASP.NET Core and Angular 2, Packt Publishing, 2016, ISBN: 978-1786465689
[5] Christian Nagel: Professional C# 6 and .NET Core 1.0, Wrox, 2016, ISBN: 978-1119096603

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 01.09.2017
Datum odevzdání: 30.04.2018



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2018 ..... Nowec

Rád bych poděkoval svému vedoucímu Ing. Michalovi Radeckému, Ph.D., za ochotu zapůjčit hardware a cenné rady při tvorbě této práce.

Abstrakt

Cílem této práce je seznámit se s možnostmi integrace služeb, představit si jejich hlavní představitele na trhu a popsat jejich silné a slabé stránky. Následně se detailněji seznámit se službou IFTTT, předvést její možnosti a praktické využití. Další část se zaměřuje na IFTTT protokol a jeho technické detaily, kterým je třeba porozumět při tvorbě aplikací, které jej chtějí využít. Poslední kapitola popisuje implementaci jednoduché webové aplikace se serverless architekturou a napojením na IFTTT pomocí webhooků.

Klíčová slova: IFTTT, iPaaS, integrační platforma jako služba

Abstract

Goal of this thesis is to inform about possibilities of service integration, then intoduce their main representatives on the market and describe their strengths and weaknesses. Next part concerns IFTTT, it's possibilities and practical application. Another part deals with IFTTT protocol and it's technical details which should be fully understood to be implemented right. Final chapter describes implementation details of a simple web application with a serverless architecture and connecting it with the IFTTT using webhooks.

Key Words: IFTTT, iPaaS, integration platform as a service

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
CLI	–	Command Line Interface
EET	–	Elektronická Evidence Tržeb
HTTP	–	Hypertext Transfer Protocol
HTTPS	–	Hypertext Transfer Protocol Secure
IFTTT	–	If This Then That
IMAP	–	Internet Message Access Protocol
IO	–	Input/Output
iPaaS	–	Integration Platform as a Service
JSON	–	JavaScript Object Notation
MS	–	Microsoft
NoSQL	–	Not only Structured Query Language
PC	–	Personal Computer
SaaS	–	Software as a Service
SMS	–	Short Message Service
SMTP	–	Simple Mail Transfer Protocol
SPA	–	Single Page Appliacation
SSL	–	Secure Sockets Layer
UI	–	User Interface
URL	–	Uniform Resource Locator

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V dnešní době jsou webové aplikace jako Facebook, Gmail, Google Drive nebo Microsoft Office 365 nedílnou součástí našich každodenních životů a to jak v práci, tak i doma. Tento software funguje sám o sobě skvěle, dokud se nenaskytne potřeba mezi službami sdílet data nebo reagovat na události. V uzavřeném ekosystému jedné firmy to ještě může fungovat dobře, ale mezi službami různých korporací zřídkakdy. V takovém případě uživatel může využít řadu aplikací třetích stran, které se specializují na propojování služeb. Mezi aplikace, které slouží jako prostředník, patří právě IFTTT, ale i řada dalších. Soubor tohoto typu služeb je nazýván iPaaS (integration Platform as a Service) [1].

Tyto služby se často zaměřují na podniková řešení, kde rychle rostou na popularitě. Umožňují rapidní integrace pomocí předpřipravených částí, které do sebe zapadají podobně jako Lego. Většina integračních služeb poskytuje vizuální zobrazení, což dále ulehčuje a zrychluje práci. Služby mnohdy běží v cloudu a tím umožňují zákazníkům integrovat služby bez potřeby nákupu dalšího hardwaru a tedy bez vysokých vstupních investic. Další výhodou spojenou s cloudem je škálovatelnost založená na aktuální potřebě. Tato podniková řešení se zaměřují především na integraci celých systému a komunikaci mezi nimi, jako třeba SAP, Oracle, Workday nebo Salesforce.

Druhou skupinou jsou Citizen iPaaS zaměřené přímo na osoby, které chtějí zautomatizovat pouze svou část práce a nemusí být tak zdatní v technologiích. Často to tedy budou třeba účetní, prodejci nebo domácí uživatelé. Také umožňují rychlý vývoj a to v uživatelsky velmi přívětivém prostředí. Typickým využitím v podnikové sféře bude přesouvání dat nebo jejich jednoduchá transformace lidmi, kteří rozumí těmto datům a dané doméně. Příkladem může být přesun odpovědí ze služby zabývající se dotazníky do tabulkového editoru. Na druhou stranu běžný domácí uživatel službu využije spíše pro automatizaci chytré domácnosti. Tato práce se zabývá právě touto skupinou služeb.

První kapitola definuje názvosloví používané v práci a rozděluje aplikace třetích stran do několika kategorií. Druhá kapitola si dává za cíl představit konkrétní integrační platformy a poté je srovnat v praktické úloze. Následně se práce zaměřuje na IFTTT a jeho praktické využití a samotný IFTTT protokol. Na závěr pak případová studie popisuje implementaci systému s návazností na IFTTT.

2 Jak funguje integrace služeb

Integrační služba neboli iPaaS zprostředkovává komunikaci mezi dalšími cloudovými službami [2]. Z uživatelského hlediska to následně vypadá tak, že ve webovém prohlížeči spojuje už hotové části systému. Tyto části lze ještě částečně nastavit pro aktuální potřeby. Až uživatel přenese svou myšlenku, tak celý proces v tom nejjednodušším případu vypadá tak, že jedna služba zareaguje na uživatelsky zajímavou událost a informuje o tom prostředníka. Ten na základě uživatelsky definovaných nastavení vyhodnotí jakou akci a na které službě vyvolat a následně tak učiní.

Obrázek 1: Proces zasílání informací

2.1 Cloudová služba

V kontextu citizen iPaaS to obvykle budou SaaS aplikace [3]. Tato aplikace poskytuje triggery a akce, které lze využívat skrze integrační službu. Komunikace mezi těmito službami probíhá obvykle pomocí HTTP protokolu a předem domluvených pravidel. Služby si můžeme rozdělit do několika kategorií podle zaměření.

2.1.1 Chytrá domácnost

Na tuto oblast se orientuje velké množství výrobců a jelikož neexistuje jednotný komunikační standard, který by dodržovali všichni výrobci, tak se integrační služba jeví jako vhodný společník. Nabízí zajímavé možnosti zejména pro technologické nadšence, kteří si chtějí zautomatizovat dům nebo byt. Není problém rozblikat žárovku, když pračka dokončí svůj program nebo zapnout topení hlasovým povelem „Ok Google, . . . “.

2.1.2 Webové aplikace

Webové služby využíváme denně, ať už se jedná o ty pro práci nebo zábavu. V každém případě si ale můžeme ušetřit spoustu času. Není problém zautomatizovat přidávání hudby do Spotify playlistu jen pomocí lajknutí videa na YouTube, archivovat historii SMS do dokumentu na Google Drive nebo vytvoření zprávy na Slacku po přijetí platby v kryptoměně. Výběr ze služeb je obrovský a v podstatě každá nabízí na výběr z několika akcí a triggerů.

Tabulka 1: Názvosloví

Český překlad	Událost	Akce	Úkol	Služba
IFTTT	Trigger	Action	Applet	Service
Zapier	Trigger	Action	Zap	App
MS Flow	Trigger	Action	Flow	Connector
Integromat	Trigger	Action	Scenario	Module

2.1.3 Mobilní zařízení

Službou se může stát i mobilní telefon s nainstalovanou aplikací dané integrační služby. Výhodou telefonu je, že jej většinou nosíme celý den u sebe, takže dokáže zprostředkovat naši polohu. Mohou také sledovat stav Wi-Fi nebo nově příchozí SMS. Některé také umožňují přidat na domácí obrazovku widget, kterým lze manuálně aktivovat applet. Akcí poté může být vytvoření notifikace, stlumení vyzvánění nebo třeba otevření webového prohlížeče na dané stránce.

2.1.4 Integrované aplikace

Do této kategorie obvykle spadají nástroje pro základní manipulaci s daty, které slouží pro přípravu dat před jejich předáním dalším službám. Především tedy parsery pro formát JSON a XML nebo naopak nástroje pro formátování dat (datum, čas, měna) do lokalizované podoby. Časovače pro pravidelné spouštění úkolu, uložení hodnoty do databáze nebo třeba odeslání emailu.

2.1.5 Webhook

Zajímavá služba určená především pro programátory a počítačově zdatnější jedince. Umožňuje v integrační službě vytvořit webovou adresu schopnou přijímat HTTP requesty. Po přijetí takového požadavku se aktivuje trigger a appletu poskytne i obsah webového požadavku. Využít HTTP protokolu lze i naopak a použít jej jako akci – tedy zaslat request na požadovanou adresu na internetu.

2.2 Názvosloví

I přesto, že každá služba má své vlastní názvosloví, tak vzhledem k tomu, že se práce zabývá především službou IFTTT, budou používány tyto pojmy skrze celou práci. Originální názvy jsou uvedeny v tabulce 1.

2.2.1 Applet

Applet je soubor triggeru a akcí, které tvoří jeden celek – jednu konkrétní úlohu, kterou chceme popsat.

2.2.2 Trigger

Trigger je vyvolán, když na službě nastane nějaká, pro nás zajímavá skutečnost. Jako příklad lze uvést službu Google Gmail a její trigger nový email. Tato událost bude dále obsahovat doplňující informace o vzniklé události.

2.2.3 Ingredience

Jsou to data, která nesou informaci o vzniklé události. Typickým příkladem budiž čas vyvolání triggeru, ale uživatelsky zajímavější data budou obsahovat například email odesilatele, předmět nebo obsah emailu.

2.2.4 Akce

Akce slouží jako reakce na událost. Obvykle využívá informace získané z události. Má také jasně dané rozhraní, které můžeme využít. Například akce služby SMS by mohla potřebovat informace jako telefonní číslo a text zprávy.

3 IFTTT a podobné služby

Ne každá integrační služba se hodí k řešení všech typů úloh. Vždy záleží na konkrétních potřebách uživatele a právě proto se tato kapitola zaměřuje na porovnání aspektů důležitých při výběru té správné služby.

3.1 IFTTT

Jedná se o nejznámější integrační službu, která cílí na extrémní jednoduchost. Po přihlášení dostaneme k výběru z několika stovek appletů. Výhodou je velká komunita okolo IFTTT, kde uživatelé vytváří další applety. Těší se také velké oblibě výrobců smart home produktů, kteří jinak integrační služby obvykle ignorují. Pro využití nabízí přes 360 služeb [4].

Vytvoření nového appletu je velice jednoduché, ale omezené. V podstatě jsme odkázáni na jeden trigger a jednu akci. Hluboko v menu lze ještě nalézt rozšířený mód Maker, který nabízí možnost vytváření složitějších appletů s podporou pro více akcí zároveň. Tyto akce ale mohou probíhat pouze paralelně a tím pádem na sebe nemohou ani nijak reagovat, takže většinou bude jednodušší si vytvořit zvlášť dva applety se stejnými triggery a různými akcemi. Další výhodou Makeru je možnost napsat si vlastní **Filter**. Je to vlastně jednoduchý skript v JavaScriptu. Nabízí podporu celého jazyka kromě IO operací, přesto je výsledný kód velmi omezený, vzhledem k minimálnímu dostupnému rozhraní jednotlivých triggerů a akcí. Rozhraní obvykle umožňuje přistupovat k ingrediencím triggerů a nastavit data akce, či akci přeskočit. Jediným pádným důvodem proč tedy vůbec využít Maker, je možnost Applet publikovat a poskytnout jej tak dalším uživatelům.

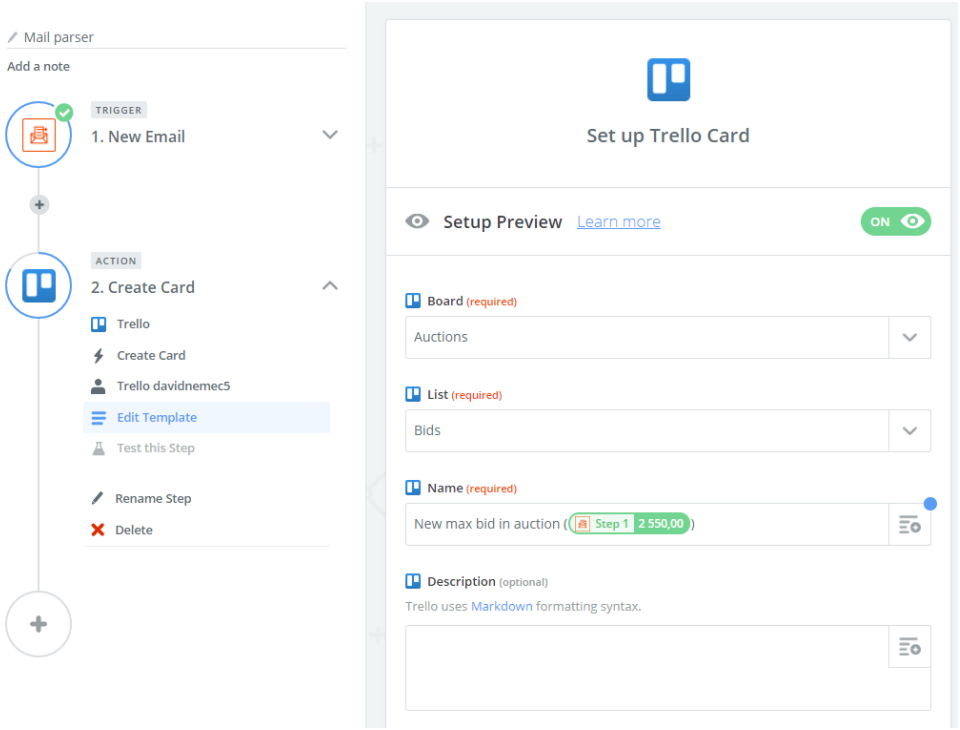
Mobilní aplikace umožňuje správu a tvorbu appletů. Dále z mobilu vytváří v podstatě další službu, což je výhodné vzhledem k tomu, že mobilní telefon máme neustále u sebe na rozdíl od počítače. Aplikace nám poté zprostředkovává události jako přijetí nové notifikace, připojení ke specifickému Bluetooth zařízení nebo WiFi. Umí také reagovat na nedostatek energie v baterii. Jako akci poté zvládne např. změnit tapetu, spustit navigaci na určité místo, vypnout vyzvánění, zapnout WiFi nebo Bluetooth. Aplikace také umožňuje přidat na domovskou obrazovku widgety, které slouží jako triggery. Jsou dostupné tři druhy. První vyvolá jen událost, ke druhému lze zadat poznámku a k poslednímu přidat nebo vyfotit obrázek.

Pokročilým uživatelům služba pro svou jednoduchost asi moc nesedne. Využití základních možností a výběr z mnoha předpřipravených appletů je více než dostačující. Výhodou je, že je naprosto zdarma a bez reklam. Zpoplatněna je jen pro firmy, které chtějí přidat novou službu do systému.

3.2 Zapier

Zapier je taktéž uživatelsky přívětivá webová aplikace. Uživatelům ale poskytuje detailnější možnosti nastavení, především díky vyspělým vestavěným službám. Samozřejmostí jsou před-

připravené applety, které si stačí vybrat a začít používat. Nabídne přes 1000 služeb třetích stran [5].



Obrázek 2: Vytvoření appletu pro parsování emailu

Prostředí pro vytvoření appletu je přehledné, ale zároveň nabízí více možností nastavení než IFTTT. V čem ale Zapier exceluje, jsou vestavěné služby. Ať už podporou emailu pomocí protokolů IMAP a SMTP, či vytvoření nové emailové schránky přímo pod doménou zapier-mail.com. Zajímavou možností je parsování emailu pomocí šablony. Vytvoření šablony probíhá tak, že uživatel přepošle email na vygenerovanou schránku zapierem a poté pouze vyznačí text, který se bude s každým emailem opakovat. Zapier také nabízí spuštění jednoduchých programů v Pythonu a JavaScriptu. Je možné vytvářet komplexnější applety díky tomu, že je umožněno skládat jednotlivé kroky tak, aby na sebe navazovaly.

```
Aktuální cena je: {{currentprice}}

Název: {{auctionname}} (číslo: {{auctionnumber}})

Přihazující: {{biddername}} ({{biddernickname}})

Konec: {{auctionend}}

S přátelským pozdravem
```

Obrázek 3: Nastavení parsování emailu

Zapier bohužel zatím nedisponuje mobilní aplikací, ale je už oznámena podpora pro systém Android [11]. Nabízí však rozšíření do prohlížeče Google Chrome, které funguje jako trigger a umožňuje uživateli jej vyvolat nebo k němu připojit i data ve formě textu.

Zapier je zdarma s omezenými možnostmi. Pro vytvoření appletu s více než dvěma kroky je třeba si už zaplatit. Některé služby, obvykle ty cílící na byznys sféru, jsou také zpoplatněny. Přidání vlastní služby je zdarma. Vývojáři mají k dispozici webové rozhraní nebo přístup přes konzoli, která nabízí více možností a vývoj pod platformou NodeJS.

3.3 Microsoft Flow

Microsoft Flow díky svým možnostem osloví především zkušenější uživatele, kteří si chtějí nakonfigurovat vše do detailu. Pro tuzemské uživatele je výhodou, že je kompletně v češtině.

Flow nabízí na výběr přes 180 služeb [6]. Applety lze dost dobře upravovat a to platí i pro ty, které vytvoříme z předpřipravených templatů. Není tedy problém se podívat „dovnitř“ připravených appletů a nechat se inspirovat. Při práci s daty si poradí s parsováním formátu JSON, filtrováním pole nebo vytvořením CSV či HTML tabulky z dat. Nabízí také podporu pro ladění appletů, kde si lze přesně prohlédnout, jak daný applet proběhl a jak vypadala data.

Microsoft nabizí také povedenou aplikaci sloužící jak pro správu appletů, tak i jako rozhraní k triggerům a akcím smartphonu. Zajímavostí je možnost přidat si na plochu widget ve tvaru tlačítka, který slouží jako trigger.

Flow od Microsoftu je pro základní použití zdarma, ale obsahuje omezení ve formě maximálního počtu spuštění 750krát měsíčně a omezeným výběrem služeb. Uživatelé využívající Office 365 dostanou 2000 spuštění měsíčně. Microsoft zde hodně cílí na byznysovou sféru a k napojení nabízí služby jako PostgreSQL, Jira, Bitbucket. Tyto služby jsou ale obvykle zpoplatněny. Nabízí také dobrou podporu dalších služeb ze svého portfolia jako Office, Dynamics, SharePoint, Azure nebo SQL Server.

3.4 Integromat

Je to služba, která nechává uživateli spoustu volnosti a lze pomocí ní vytvořit i velmi složité applety. Kromě toho ale samozřejmě nabízí i ty předdefinované, které lze využít i jako inspiraci, protože je lze okopírovat a upravit. Jelikož se jedná o českou firmu, tak nabízí i českou mutaci a podporu pro další české služby – především banky, ale třeba také EET.

Zajímavou funkcí, kterou Integromat nabízí, je agregace dat, která umožňuje se skupinou dat pracovat jako s jednou položkou. Na druhou stranu lze přes data také iterovat. Příkladem může být událost příchozí email obsahující přílohy, které lze pomocí iterace zpracovat každou zvlášť. Nabízí bohaté možnosti pro práci s daty podobné těm z Excelu a obsahuje také funkce pro zpracování dat získaných z webu, tedy parsování JSONu a XML. Aktuálně zprostředkovává komunikaci mezi více jak 200 službami [7].

Mobilní aplikace postrádá možnost úpravy scénářů, čemuž se vzhledem ke složitosti nelze divit. Slouží tedy jen jako rozhraní, kterým lze z Integromatu přistupovat k SMS, notifikacím, vytočit číslo, či otevřít webovou stránku.

Integromat je zdarma pro omezený počet operací a množství přenesených dat. Seznam podporovaných služeb je rozsáhlý, pokud ale požadovanou službu nenalezneme a podporuje autorizaci například pomocí protokolu OAuth2 a má webové API, tak uživatel má možnost si funkcionalitu přidat sám. Mezi nabízenými službami najdeme i přímé napojení na databázové servery jako třeba MongoDB nebo PostreSQL. Poskytuje také kvalitní logging všeho, co se děje a následné hledání chyb je jednoduché a přehledné. Podle podpory pro EET, fakturovací služby a přímé napojení na databázi lze vypozorovat, že se Integromat hodně zaměřuje na firemní sféru. Pro použití nabízí 1000 operací měsíčně zdarma, kde jedna operace zhruba odpovídá jedné akci nebo události.

3.5 Porovnání služeb v praxi – Web API

Pro porovnání služeb si vytvoříme applet, který v praxi využijeme při těžbě kryptoměn. Problém je takový, že těžíme Ethereum[1] a chceme dostat notifikaci na telefon, když stanice bude offline nebo mít vysoký počet stale shares. Mining pool, který využijeme, má k dispozici API s daty ve formátu JSON. Ve zkratce tedy chceme v pravidelných intervalech stahovat JSON, následně jej rozparsovat a na základě dat se rozhodnout, zda poslat notifikaci. Toto vše chceme zdarma a v co nejkratších intervalech. V této kapitole se sice zaměříme na kryptoměny, problém ale lze zobecnit na jakékoliv jiné API.

V dokumentaci poolu[2] najdeme API, které vyhovuje našim požadavkům[3].


```
{
  "status": "OK",
  "data": {
    "time": 1524313200,
    "lastSeen": 1524313110,
    "reportedHashrate": 94447207,
    "currentHashrate": 100666666.66666667,
    "validShares": 88,
    "invalidShares": 0,
    "staleShares": 4,
    "averageHashrate": 94234182.09876543,
    "activeWorkers": 1,
    "unpaid": 51840220675201390,
    "unconfirmed": null,
```


[1] https://www.ethereum.org/
[2] https://ethermine.org/api/pool
[3] https://api.ethermine.org/miner/7bb65094af10c48049055737c0c290300da9e84b/currentStats

```
    "coinsPerMin": 0.000004961687716548715,
    "usdPerMin": 0.0029225829156786895,
    "btcPerMin": 3.3471545335837635e-7
  }
}
```

Výpis 1: Odpověď ve formátu JSON z Ethermine

3.5.1 IFTTT

Mezi nabízenými akcemi není žádná, pro zaslání HTTP GET requestu. Tím pádem nemáme jak získat data a nemůžeme tuto službu nijak využít.

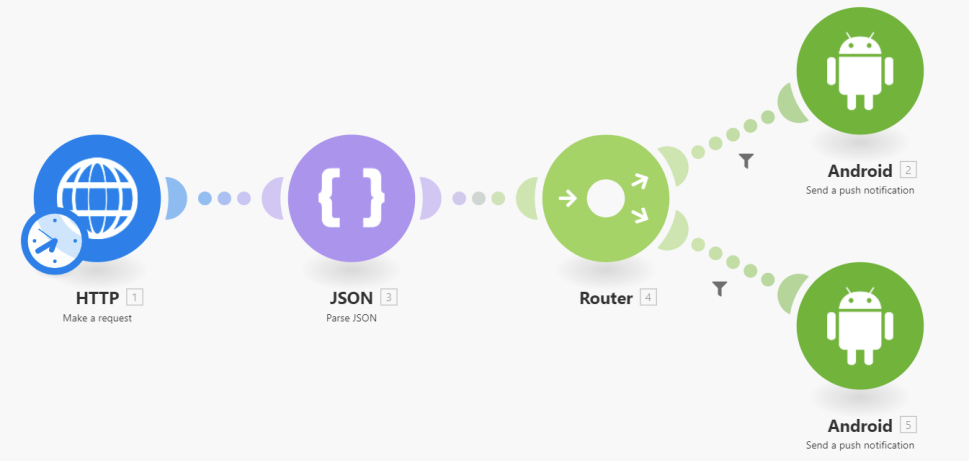
3.5.2 Zapier

Vytvoříme nový applet a jako trigger přidáme **Schedule** a nastavíme jej na aktivaci každou hodinu. Následující akci zvolíme **Webhooks** – **GET** a nastavíme adresu API. Jelikož Zapier nemá svou mobilní aplikaci, tak si pro zasílání notifikací zvolíme **Pushbullet** a jeho akci **Send a Note**. Dále také chybí prostředek pro rozdělení do více souběžných větví, takže nemůžeme zasílat dva typy notifikací.

3.5.3 Integromat

V novém appletu začneme nastavením intervalu, jak často by se měl applet provádět, na 15 minut. Jako akci nastavíme **HTTP** – **Make a request** s metodou GET a URL potřebného API. Dále pokračujeme přidáním modulu **JSON** s akcí **Parse JSON** a nastavíme parsování ingredience `data` z HTTP requestu. Následně musíme manuálně spustit applet, aby Integromat zjistil, co požadovaný JSON obsahuje a mohl dále tyto části nabídnout jako ingredience. Jako další krok přidáme **Router**, který umožní rozdělit datový tok na více částí. Na oba konce přidáme akci **Android** – **Send a push notification**. Nastavíme zařízení, nadpis notifikace a jako akci na klik otevření URL adresy dashboardu[4]. Před každou notifikací ještě nastavíme filtr s podmínkou `activeWorkers < 2` a `staleShares >= 10`.

[4] https://ethermine.org/miners/7bb65094af10c48049055737c0c290300da9e84b/dashboard



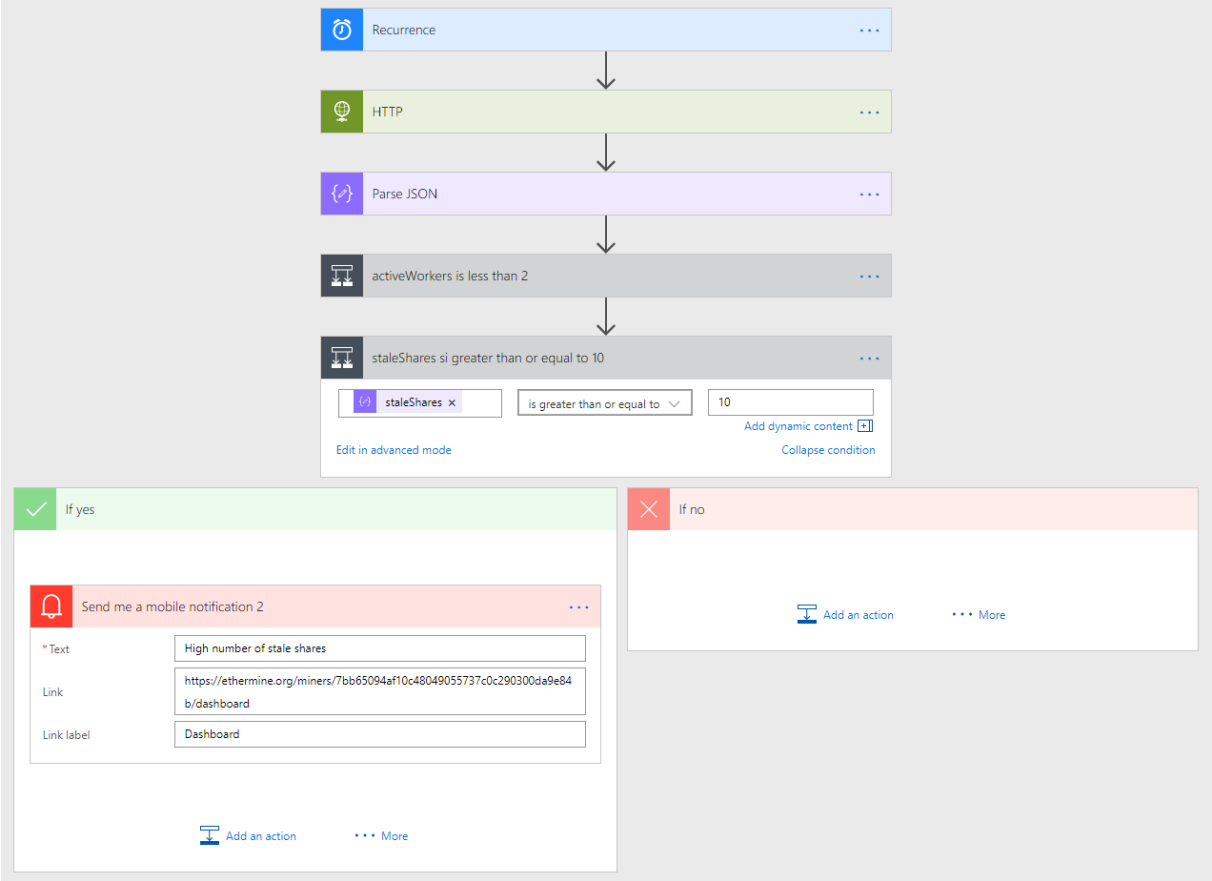
Obrázek 4: Výsledný applet pro pravidelnou kontrolu API

Jedinou nevýhodou je, že i s takto jednoduchým appletem se brzo vyčerpá limit 1000 operací zdarma. Přitom pokud bychom chtěli kontrolovat API co nejčastěji, ideálně každou čtvrthodinu, tak to měsíčně vyjde na 5760 operací[5].

3.5.4 Microsoft Flow

Začneme vytvořením appletu a přidáme si trigger **Schedule – Recurrence** nastavený na aktivaci appletu každých 15 minut. Následně zvolíme akci **HTTP** typu GET na adresu API. Pokračujeme akcí **Parse JSON**, která pro správnou funkci vyžaduje schéma parsovaného JSONu. Je to v podstatě jen další JSON, který obsahuje informace o názvech a typech hodnot. Naštěstí schéma nemusíme tvořit ručně, ale lze si ho nechat vygenerovat. Přidáme ještě dva bloky s podmínkami stejnými jako v případě Integromatu a pro kladné větve zadáme a nastavíme akce **Send me a mobile notification**.

[5]Při optimistickém předpokladu, že se nevyskytnou žádné problémy a vždy proběhnou pouze bloky HTTP a JSON a vezmeme v potaz měsíc o 30 dnech ($2 \cdot 4 \cdot 24 \cdot 30 = 5760$)



Obrázek 5: Výsledný applet pro kontrolu API v časovém intervalu

Pokud bychom chtěli applet spouštět každou čtvrt hodinu, tak to znamená 2880 spuštění měsíčně[6].

3.5.5 Shrnutí

Ve verzi zdarma není pro častou kontrolu vyhovující žádná aplikace. Počet spuštění v Microsoft Flow pro kontrolu vystačí pouze na hodinové intervaly. V případě Integromatu je to dokonce hodina a půl a to pouze v ideálním případě. Vzhledem k tomu, že Zapier ve verzi zdarma nabízí jen applety o dvou krocích, ale potřeba jsou minimálně tři, tak v této úloze neuspěje. IFTTT pro tento úkol nemá cenu uvažovat.

[6] $4 \cdot 24 \cdot 30 = 2880$

Tabulka 2: Přehled služeb

Název	Zdarma	Počet služeb	Editace appletu v aplikaci	Aplikace jako služba
IFTTT	Ano	360	Ano	Ano
Zapier	100 spuštění/měsíc	1000	-	-
MS Flow	750 spuštění/měsíc	180	Ano	Ano
Integromat	1000 operací/měsíc	200	Ne	Ano

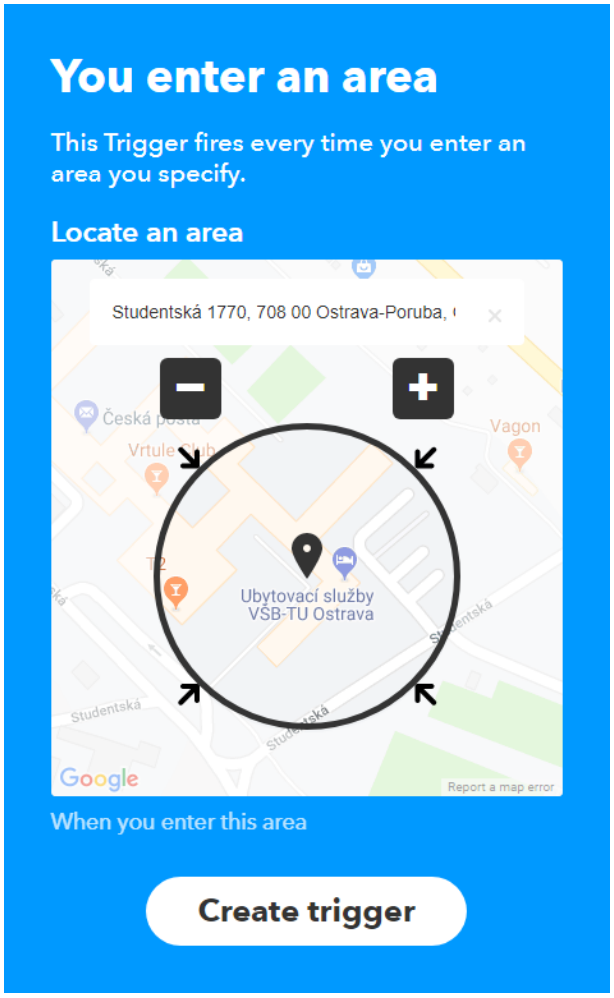
4 Možnosti a využití IFTTT

V této kapitole si předvedeme, jak jednoduché je vytváření appletů ve službě IFTTT.

4.1 Zapnutí zásuvky při příchodu domů

Vytvoření jednoduchého appletu, který můžeme využít v případě potřeby zapnutí spotřebiče těsně před tím, než přijdeme domů.

1. V menu vybereme **My Applets** → **New Applet**
2. Klikneme na **+this**
3. Ze seznamu vybereme **Location**
4. Vybereme si trigger **You enter an area**
5. Zvolíme požadovanou lokaci



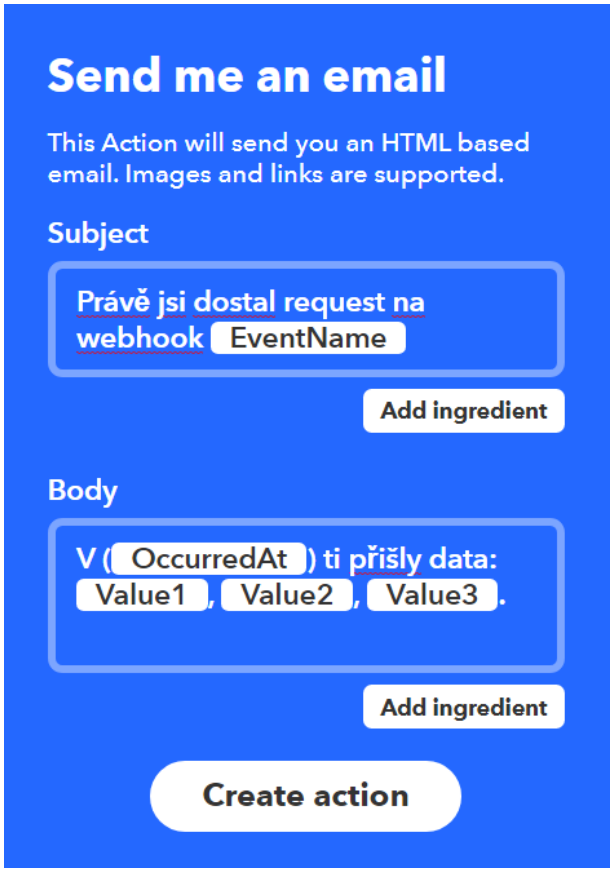
Obrázek 6: Vybrání lokace

6. Klikneme na **+that**
7. Vybereme **D-Link Smart Plug**
8. Zvolíme akci **Turn on**
9. Vybereme správnou zásuvku ze seznamu
10. Hotovo, applet můžeme aktivovat a otestovat

4.2 Pošli email po přijetí HTTP požadavku

Applet pro demonstraci použití HTTP webhooku, který může být užitečný například pro připojení nějakého skriptu s IFTTT.

1. V menu vybereme **My Applets** → **New Applet**
2. Klikneme na **+this**
3. Ze seznamu vybereme **Webhooks**
4. Vybereme si trigger **Receive a web request**
5. Event pojmenujeme například `send_email` (tento název je součástí URL, na kterou se zasílají web requesty)
6. Klikneme na **+that**
7. Vybereme **Email**
8. Zvolíme akci **Send me an email**
9. Nastavíme šablonu podle našich představ. Webhook trigger nabízí 5 hodnot (ingrediencí), které můžeme využít:
    - `EventName` - název eventu, který jsme zadali dříve
    - `OccuredAt` - čas, kdy dorazil web request
    - `Value1-3` - hodnoty, které můžeme poslat ve web requestu JSONem, lze zasílat pouze text, cokoliv jiného se převede opět na text

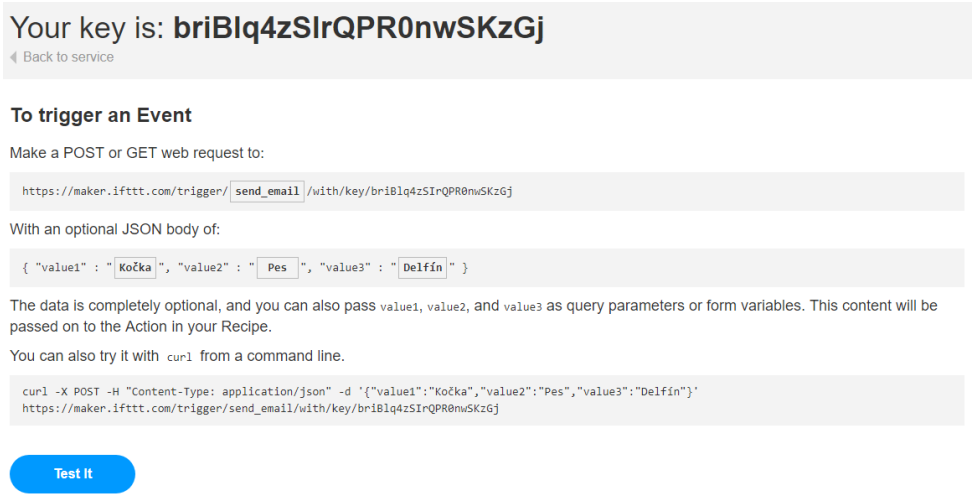


Obrázek 7: Nastavení akce zaslání emailu

10. Potvrdíme a tím jsme vytvořili náš první applet. Potřebujeme ještě získat adresu, na kterou budeme zasílat HTTP požadavky.

11. V menu vybereme **My Applets** → **Services**

12. Vyhledáme **Webhooks** a otevřeme dokumentaci



Obrázek 8: Získání webhook adresy

13. Tímto jsme získali potřebný klíč a URL, na které můžeme otestovat, zda vše funguje.

5 Technické detaily IFTTT

Pro implementaci vlastní služby je třeba dodržovat zásady IFTTT protokolu. Tento protokol byl vytvořen, aby vývojářům poskytnul jasné a stručné rozhraní, jak uskutečnit komunikaci.

5.1 Obecné požadavky rozhraní

IFTTT protokol říká, že veškerá komunikace v produkčním prostředí by měla probíhat šifrovaným protokolem HTTPS. Určuje také, jak budou vypadat URL adresy. Každá adresa určena pro využití s IFTTT by měla mít API prefix následovaný informací, že se jedná o IFTTT protokol v určité verzi.

```
https://api.myservice.com/ifttt/v1/triggers/comment_updated
https://myservice.com/api/ifttt/v1/actions/post_comment
```

Výpis 2: Adresy trigger endpointů

IFTTT definuje také formu hlavičky odpovědi na HTTP dotaz. Zpráva musí využívat kódování znaků formou UTF-8, jež je dnes nejrozšířenější způsob kódování textu na webu [10]. Musí také podporovat kompresi a tělo odpovědi zasílat ve formátu JSON. Úspěšná odpověď musí obsahovat položku `data` a hodnotou může být objekt či pole objektů. Neúspěšná odpověď bude místo dat obsahovat pole objektů chyb `errors`. Každý chybový objekt bude následně uchovávat řetězec `message` s chybovou hláškou.

```
{
  "data": {
   "timestamp": 1517255367753,
   "note": "Pracovat!"
  }
}
```

Výpis 3: Tělo úspěšné odpovědi naší služby ve formátu JSON

```
{
  "errors": [
    {
      "message": "Sorry jako!"
    }
  ]
}
```

Výpis 4: Tělo neúspěšné odpovědi naší služby ve formátu JSON

Tabulka 3: Stavové kódy pro odpověď služby

Stavový kód	Popis využití
200	Požadavek byl úspěšný
400	Chybný požadavek, je třeba dále specifikovat v čem byl problém
401	Neplatný autentizační token
404	Požadovaná adresa nenalezena
500	Chyba v logice aplikace
503	Aplikace dočasně nedostupná

Při implementaci bychom neměli provádět striktní validaci dat ze strany IFTTT, jelikož se v budoucnosti počítá s jejich rozšiřováním. Může se jednat o užitečné informace navíc, nepovinné prvky nebo data pro usnadnění ladění.

5.2 Autentizace

Pokud chceme přidat službu, která vyžaduje uživatelské účty, je třeba naimplementovat autentizační endpoint. Pro autentizaci se využívá otevřeného standardu OAuth2. Je to autorizační framework, který umožňuje aplikacím třetích stran získat omezený přístup ke službě bez potřeby získat uživatelské heslo. Služba s potvrzením uživatele vydá token, který poté slouží k přístupu ke chráněným datům. Výhodou je, že tyto tokeny mohou být nastaveny tak, aby povolily přístup jen k určité podmnožině dostupných informací, což zvyšuje bezpečnost [9].

Tak jako OAuth2, tak i IFTTT protokol podporuje 2 typy tokenů - access token a refresh token. Hlavním rozdílem mezi nimi je, že refresh tokeny se používají jen pro komunikaci s autentizačním serverem a nikdy neslouží pro přístup k datům, zatímco access tokeny se mohou používat v obou případech.

5.2.1 Access token

Access tokeny slouží pro autentizaci a jako zdroj identity. Jeden token by tedy měl odpovídat jednomu uživateli služby. Aby se uživatelé nemuseli často znovu přihlašovat, tak je třeba vydávat tokeny bez expirace. Proces autentizace a autorizace probíhá v několika krocích.

1. Uživatel se rozhodne použít službu, ke které je třeba se přihlásit, tak jej IFTTT přesměruje na OAuth2 autorizační adresu služby. Potřebné informace se přenesou v URL jako parametry query stringu.

   Parametry:

   - `client_id` unikátní identifikátor naší služby, který si volíme sami v nastavení, ideálně to může být název firmy nebo produktu
   - `response_type` defaultní hodnota je `code`

- `scope` rozsah API, ke kterému je udělen přístup, defaultní hodnotou je `ifttt`, což znamená přístup ke každému triggeru a akci. Lze nastavit vlastní rozsah. Výhodou nastavení přístupu ke všemu je, že uživatel se nebude muset opakovaně přihlašovat pro využití různých triggerů a akcí
- `state` token, sloužící pro potvrzení, že se nejedná o podvržený požadavek
- `redirect_uri` adresa, kam přesměrovat uživatele po vykonání požadavku

Přesto, že IFTTT poskytuje adresu pro přesměrování, tak je doporučeno použít autorizační adresu naší služby na IFTTT

```
https://ifttt.com/channels/{{service_id}}/authorize
```

Výpis 5: Adresa autentizačního endpointu

2. Uživatel je přesměrován zpátky na web IFTTT, kde se v query stringu přenese speciální kód nebo v případě chyby její informace. Tím, že se IFTTT vrátí stejný `state`, tak pozná, že se nejedná o podvrh.

   Parametry:

   - `code` vygenerovaný kód
   - `state` token, sloužící pro potvrzení, že se nejedná o podvržený požadavek
   - `error` chybový kód, například `access_denied`

3. Pokud nenastala žádná chyba, tak proběhne výměna kódu za access token případně i refresh token. To se provede HTTP POST požadavkem na OAuth2 token adresu

   Parametry:

   - `grant_type` defaultně `authorization_code`
   - `code` kód vygenerovaný v minulém kroku
   - `client_id` unikátní identifikátor naší služby, který si volíme sami v nastavení, ideálně to může být název firmy nebo produktu
   - `client_secret` v podstatě „heslo“ naší služby, slouží pro zabezpečení komunikace
   - `redirect_uri` adresa, kam přesměrovat uživatele po vykonání požadavku, doporučeno nepoužívat

   Pokud autorizační kód není validní, je třeba vrátit JSON s chybovou hláškou pod klíčem `error` a chybovým kódem 401

   V kladném případě bude stavový kód 200 a vrácený JSON obsahovat hodnoty `token_type`, `access_token` případně i `refresh_token`.

```
{
  "token_type": "Bearer",
  "access_token": "0984c90c16bb4e5dbdb8db215f402294"
}
```

Výpis 6: Tělo úspěšné odpovědi s access tokenem

5.2.2 Refresh token

Refresh tokeny slouží pro posílení bezpečnosti. Dovolí nám to nastavit access tokenům krátkou životnost v řádu minut až hodin. Pokud by došlo ke kompromitaci access tokenu, tak útočník má velmi omezenou dobu, kdy je schopen napáchat škody. Po expiraci je totiž token znovu nepoužitelný a je třeba využít refresh token pro získání nového access tokenu. Pro zachování pohodlí uživatelů by refresh token měl být vydán bez expirace, poté nebude uživatel nucen se opakovaně přihlašovat pomocí hesla.

Pokud se rozhodneme využít refresh tokenu, musíme změnu zavést také do nastavení v IFTTT. Po tomto kroku bude komunikace mezi naší službou a IFTTT stále využívat access tokeny, ale v případě expirovaného nebo jinak neplatného tokenu je třeba vrátit chybový stavový kód 401, na což IFTTT zareaguje požadavkem na token refresh endpoint, vyžádá si nový access token a poté zopakuje původně zamýšlenou akci.

Požadavek na refresh endpoint bude typu POST s následujícími parametry:

- `grant_type` defaultně `refresh_token`
- `client_id` unikátní identifikátor naší služby, který si volíme sami v nastavení, ideálně to může být název firmy nebo produktu
- `client_secret` v podstatě „heslo“ naší služby, slouží pro zabezpečení komunikace
- `refresh_token` token získaný při autentizaci

Odpovědí bude JSON obsahující `access_token`

5.2.3 Kontrola stavu uživatele

Pro dodržení požadavků, jak si je zvolilo IFTTT, je třeba také naimplementovat endpoint, který bude poskytovat informace o uživateli. V těle odpovědi musí být název účtu, unikátní identifíkátor a nepovinně také URL ke konfiguraci uživatelského účtu na naší službě. IFTTT tento endpoint poprvé využije po přihlášení, kdy si zjistí základní informace. Jelikož je pro přístup k datům potřeba autentizace, tak si IFTTT zvolilo tento endpoint pro otestování stavu autentizace uživatele. Z tohoto důvodu jej IFTTT může občas kontaktovat.

```
Authorization: Bearer {{token}}
Accept: application/json
Accept-Charset: utf-8
Accept-Encoding: gzip, deflate
Content-Type: application/json
X-Request-ID: {{random_uuid}}
```

Výpis 7: Hlavička požadavku na autentizovaný endpoint služby

```
IFTTT-Service-Key: {{ifttt_service_key}}
Accept: application/json
Accept-Charset: utf-8
Accept-Encoding: gzip, deflate
X-Request-ID: {{random_uuid}}
```

Výpis 8: Hlavička požadavku na neautentizovaný endpoint služby

Hlavička požadavku musí vždy obsahovat `IFTTT-Service-Key` pokud se jedná o endpoint vyžadující autentizaci. Volitelně lze zasílat `X-Request-ID`, což je unikátní identifikátor requestu. Jeho hodnota se loguje a může být cenným pomocníkem při hledání chyb.

5.3 Triggery

Podle IFTTT protokolu každý trigger musí mít odpovídající endpoint. Data tohoto endpointu nejsou tvořeny ničím jiným než seznamem až padesáti již proběhnutých událostí. IFTTT bude trigger endpoint v pravidelných intervalech dotazovat a porovnávat data. Pokud narazí na novou událost, tak ji zpracuje a uživatelům na základě tohoto triggeru se provede určená akce. Typ tohoto způsobu dotazování nazýváme polling.

Každý trigger endpoint by měl vracet v základním nastavení 50 posledních událostí, ať už byly zobrazeny nebo ne a musí být seřazeny sestupně podle data vytvoření. Každá událost musí obsahovat údaj o čase v Unixovém formátu v sekundách, unikátní identifikátor a jeden field pro každou ingredienci. Počet událostí může být specifikován ze strany IFTTT parametrem `limit` v dotazu. Toto nastavení je potřebné pro případ, kdy by počet nových událostí přesáhl základní limit, aby IFTTT mohlo reagovat na všechny nové události. Dotazování probíhá každých 15 minut.

Endpoint může ale nemusí podléhat autentizaci, záleží na podstatě dat. V prvním případě musí request směrem na naši službu obsahovat hlavičku `Authorization` a odpovídající token. V druhém případě IFTTT zašle v hlavičce `IFTTT-Service-Key`. Jedná se o klíč, který je unikátní pro každou službu a lze jej nalézt v nastavení na webu IFTTT.

```
{
```

```
  "data": [
    {
      "comment": "Ahoj svete!",
      "isPublic": true,
      "section": "49a48aeb-ac47",
      "meta": {
        "id": "8a083cc3-ae2d-4f07-9831-c0e86ae578bf",
        "timestamp": "1516041045"
      }
    },
    {
      "comment": "Caute lidi",
      "isPublic": false,
      "section": "8d33b03a-4514",
      "meta": {
        "id": "a641f506-85d7-4446-a542-528cb3fb4fb2",
        "timestamp": "1515954645"
      }
    }
  ]
}
```

Výpis 9: Tělo odpovědi při pollingu trigger endpointu

5.3.1 Realtime API

Systém pollování nových událostí není příliš vhodný, pokud potřebujeme, aby se k uživateli dostala nová událost co nejdříve. Proto IFTTT podporuje i tzv. Realtime API, které umožňuje naší službě oznámit vznik nové události. IFTTT následně udělá požadavek na normální Trigger API a stáhne aktuální data. Při využití této možnosti bude IFTTT pollovat naší službu s větším časovým odstupem než 15 minut, což razantně sníží datový provoz, pokud nebudou dostupná žádná nová data. Naopak v případě aktualizace se data mohou zpracovat téměř okamžitě a uživatel nemusí čekat.

Pro notifikování o nových událostech je třeba zaslat HTTP POST request pro každou službu na jednotnou adresu `https://realtime.ifttt.com/v1/notifications`. Tělo požadavku musí obsahovat pole `data`, jež obsahuje `user_id` nebo `trigger_identity`. Využití `user_id` je neefektivní jak pro naši službu, tak pro IFTTT. Způsobí totiž pollování všech trigger endpointů i v případě, že k žádné změně nedošlo. Zatímco využitím `trigger_identity` oznámíme IFTTT, aby zkontrolovalo jen relevantní trigger endpointy. V jedné notifikaci můžeme IFTTT oznámit změnu maximálně 1000 událostí.

```
{
  "data": [
    {
      "user_id": "1d85e8b8"
    },
    {
      "trigger_identity": "1eec4679c9424b529b72f783f58d9348"
    }
  ]
}
```

Výpis 10: Tělo požadavku notifikace Realtime API

5.3.2 Trigger identity

Jedná se o unikátní identifikátor triggeru a jeho nastavení a IFTTT jej používá v každém requestu na endpoint triggeru. Pokud by tedy uživatel používal stejný trigger se shodným nastavením ve více appletech, tak bude mít pokaždé stejný identifikátor.

Představme si, že vytváříme službu pro chytrou domácnost. Máme chytrý zvonek a chceme k němu vytvořit trigger "Někdo zvoní". Také ale chceme minimalizovat počet požadavků na naše API a tak zvonek defaultně nebude zasílat žádné informace do cloudu. Pak se ale náš uživatel rozhodne využít tohoto triggeru a po vytvoření appletu, IFTTT zavolá trigger endpoint spolu s informací o `trigger_identity`. Naše služba si toto ID zapamatuje a oznámí zvonku, aby začal zasílat informace o každém zazvonění do cloudu.

Pokud by se uživatel rozhodl nepoužívat tento trigger, tak se to naše služba nemá jak dozvědět. Můžeme ale implementovat endpoint, na kterém nám IFTTT tuto informaci sdělí. IFTTT zašle DELETE request s názvem triggeru, `trigger_identity` a v těle prázdný JSON. My jsme nyní podle těchto informací schopni identifikovat zvonek, na kterém deaktivujeme zasílání informací. Implementace tohoto endpointu není povinná, ale při využití realtime API často dává smysl jej využít. Tento endpoint bude ve tvaru
`/ifttt/v1/triggers/{{trigger}}/trigger_identity/{{trigger_identity}}`.

5.3.3 Data triggeru

Triggery umožňují definovat nastavení, které uživatel musí zadat. Toto nastavení může být ve formě volného textu nebo výběru z možností. Pro využití výběru, neboli **dynamic options** jak jsou nazvány v IFTTT protokolu, musíme implementovat endpoint s URL ve tvaru
`/ifttt/v1/triggers/{{trigger}}/fields/{{field}}/options`

Odpovědí bude JSON s polem dvojic názvu a hodnoty. V případě, že uživateli chceme nabídnout větší množství možností na výběr, může dávat smysl tyto možnosti seskupit do kategorií

a pro uživatele je tak vizuálně oddělit a usnadnit mu výběr. Kategorii vytvoříme nahrazením hodnoty vnořeným polem. Pro uživatele platí, že kategorii nelze vybrat, ale pouze hodnoty v ní.

```
{
  "data": [
    {
      "label": "Delfin",
      "value": "78805c4b"
    },
    {
      "label": "Orangutan",
      "value": "ab33d428"
    },
    {
      "label": "Kocky",
      "values": [
        {
          "label": "Britska kratkosrsta kocka",
          "value": "4e2fce88"
        },
        {
          "label": "Perska kocka",
          "value": "77a01a6b"
        }
      ]
    }
  ]
}
```

Výpis 11: Tělo odpovědi na dynamic options

Pokud chceme využít pro vstup textové pole, ale nechceme uživateli povolit zadat jakýkoliv text, tak musíme tento vstup zvalidovat. Vytvoříme si tedy endpoint ve tvaru `{{prefix}}/ifttt/v1/triggers/{{název}}/fields/{{název_fieldu}}/validate` a IFTTT jej bude volat s jednou hodnotou v těle JSONu. Odpovědí by měl být vždy status code 200, kde v těle JSONu se bude nacházet, zda je field validní. V negativním případě je možno ještě přidat zprávu o chybě.

```
{
  "data": {
    "valid": false,
    "message": "Sorry, no items found."
```

```
  }
}
```

Výpis 12: Tělo negativní odpovědi na validaci fieldu

Za normálních okolností jsou validační endpointy volány vždy samostatně. Pokud ale potřebujeme zvalidovat inputy v kontextu, tak IFTTT nabízí tzv. kontextuální validaci. V praxi jde vlastně o to, že se všechna políčka odešlou pro zvalidování najednou v jednom požadavku. V případě využití tohoto typu validace nemusíme implementovat kontrolu pro jednotlivé fieldy zvlášť, jelikož je IFTTT nikdy nebude kontaktovat.

```
{
  "values": {
    "imageName": "Roztomile kotatko",
    "imageUrl": "www.someservice.url/pic"
  }
}
```

Výpis 13: Tělo požadavku na kontextuální validaci

```
{
  "data": {
    "imageName": {
      "valid": true
    },
    "imageUrl": {
      "valid": false,
      "message": "Sorry, no image found at URL: \"www.someservice.url/pic\"."
    }
  }
}
```

Výpis 14: Tělo odpovědi na kontextuální validaci

5.4 Akce

Stejně jako triggery, tak i každá akce musí mít svůj vlastní endpoint. IFTTT na tento endpoint zašle `HTTP POST` dotaz jen v případě, že si přeje vyvolat danou akci dle nějakého appletu. Opět platí, že endpoint může, ale nemusí vyžadovat autorizaci. IFTTT zašle požadavek s těmito parametry:

- `actionFields` objekt, kde každý klíč představuje název fieldu a obsahuje jeho hodnotu

- `user` objekt obsahující informace o uživateli
- `ifttt_source` nepovinný objekt, obsahující unikátní identifikátor uživatelského appletu a URL odkazující na něj. Na tuto adresu bude moci přistoupit pouze vlastník, jelikož uživatelské applety jsou soukromé.

```
{
  "actionFields": {
    "title": "Vytrit podlahu",
    "date": "2018-04-15T20:58:55+02:00"
  },
  "ifttt_source": {
    "id": "1",
    "url": "https://ifttt.com/myrecipes/personal/1"
  },
  "user": {
    "timezone": "Central European Summer Time"
  }
}
```

Výpis 15: Tělo požadavku na endpoint akce

V těle odpovědi se bude nacházet pole `data`, obsahující právě jeden objekt s `id`, které identifikuje vytvořená nebo modifikovaná data. Dále může obsahovat `url` odkazující na tato upravovaná data.

```
{
  "data": [
    {
      "id": "234325",
      "url": "http://myservice.com/tasks/1523730645"
    }
  ]
}
```

Výpis 16: Tělo odpovědi na endpoint akce

5.4.1 Přeskočení akce

Pokud nastane chyba, tak se IFTTT vynasnaží o opakování akce opětovnými požadavky na endpoint akce. Počet opakování není blíže specifikován, ale po jejich vyčerpání bude akce přeskočena.

Pokud si jsme jisti, že za žádných okolností nejsme schopni splnit požadavek, můžeme dát pokyn pro přeskočení akce ihned. Jen je třeba vrátit chybový stavový kód 400 a blíže specifikovat chybu, která se následně zobrazí uživateli.

Příkladem pro tento typ přeskočení by mohla být například situace, kdy chceme provést nahrání obrázku na Google Photos, ale soubor by nebyl ve vhodném formátu.

```
{
  "errors": [
    {
      "status": "SKIP",
      "message": "Image is not in correct format"
    }
  ]
}
```

Výpis 17: Tělo odpovědi s indikací přeskočení akce

5.4.2 Data akce

Stejně jako triggery tak i akce mohou obsahovat **dynamic options**. Nastavení je shodné jako u triggerů 5.3.3 s tím rozdílem, že akce nepodporují dynamickou validaci.

5.5 Status služby

Aby IFTTT mělo přehled o celkovém stavu naší služby, musíme implementovat status endpoint, který tyto informace zprostředkuje. Tento endpoint není uživatelský a proto nesmí vyžadovat access token. Adresa endpointu by měla být ve tvaru `{{api_url_prefix}}/ifttt/v1/status` a bude na něj dotazováno metodou GET. Vyžadované odpovědi jsou pouze dvě. Status code 200 v případě, že je všechno v pořádku. Pokud je služba nedostupná, tak status code 503. Tělo neobsahuje žádná další data.

5.6 Testování a publikace

Před tím, než publikujeme službu pro širou veřejnost, musíme vše řádně otestovat. IFTTT nám v tomto ohledu vychází vstříc a nabízí automatický testovací nástroj. Pokud těmto testům naše služba vyhoví, tak můžeme požádat o kontrolu zaměstnanců IFTTT. V případě, že i tímto krokem projdeme, tak už nám nic nebrání službu publikovat.

5.6.1 Testovací endpoint

Pro spuštění testů musíme implementovat speciální endpoint, který bude sloužit pro notifikaci naší služby, že IFTTT se chystá testovat. Adresa bude ve tvaru `{{api_url_prefix}}/ifttt/v1/test/setup`.

Odpovědí tohoto endpointu budou data, která IFTTT využije jako vstupy pro testované triggery a akce. Dále by si služba měla připravit smysluplná data pro odpovědi na testovaných endpointech.
Odpověd na metodu POST bude obsahovat následující parametry:

- `acessToken` platný token, který bude IFTTT používat pro endpointy vyžadující autentizaci
- `triggers` všechny triggery, které služba nabízí. Pokud trigger obsahuje fieldy, tak je nutno je poskytnout
- `triggerFieldValidations` pokud daný trigger poskytuje dynamickou validaci, tak pro něj musíme poskytnou platnou a neplatnou možnost
- `actions` všechny akce, které služba nabízí. Pokud akce obsahuje fieldy, tak je ntuno je poskytnout
- `actionRecordSkipping` pokud daná akce poskytuje přeskočení akce, tak můžeme zadat hodnoty, které vyvolají přeskočení

IFTTT při testování trigger endpointu očekává zobrazení alespoň tří událostí.

```
{
  "data": {
    "accessToken": "373af7b5726a4e0fb160bf6f4ca03dc2",
    "samples": {
      "triggers": {
        "comment_updated": {
          "forum": "IT a technologie"
        }
      },
      "triggerFieldValidations": {
        "comment_updated": {
          "forum": {
            "valid": "IT a technologie",
            "invalid": "Forum, jehoz nazev urcite neexistuje"
          }
        }
      },
      "actions": {
        "post_comment": {
          "title": "IFTTT nebo Integromat?",
```

```
        "text": "Je lepsi IFTTT nebo integromat?"
      }
    },
    "actionRecordSkipping": {
      "post_a_photo": {
        "title": "IFTTT nebo Integromat?",
        "text": ""
      }
    }
  }
}
```

Výpis 18: Tělo odpovědi testovacího endpointu

6 Případová studie

Cílem práce je mimo jiné vytvořit webovou aplikaci s napojením na IFTTT a zásuvku D-Link DSP-W215. Aplikace je pojata obecněji a umožní uživateli spouštět různé applety z jednotného rozhraní ve webovém prohlížeči. Tato kapitola popisuje použité technologie a postup při vývoji.

Obrázek 9: Chytrá zásuvka typu D-Link DSP-W215

6.1 Použité technologie

Rozhodl jsem se využít serverless architektury, která se soustředí na programování samotných endpointů a funkcí místo toho, abych se musel zabývat implementací celého serveru pod některou z tradičních platforem, jako je např. ASP.NET. Tento přístup umožňuje velmi rychlý vývoj. Konkrétně jsem se rozhodl použít Google Firebase[7] a řadu jejich služeb jako vlastní NoSQL databázi Firestore, Cloud Functions, modulu pro autentizaci a hosting včetně SSL certifikátu [12].

Pro implementaci samotného webu jsem vybral framework ReactJS, který je vhodný pro tvorbu SPA[8]. Používám zde také npm[9] package `firebase`, který umožňuje jednoduše používat funkce Firebasu. Pro vytvoření základní kostry aplikace jsem využil package `create-react-app`[10], který programátora naprosto odstiňuje od konfigurace potřebné pro build aplikace. V projektu je také balíček `material-ui`[11] obsahující hotové react komponenty implementující Material Design společnosti Google.

[7] https://firebase.google.com/

[8] Webová aplikace, která mění obsah stránky pomocí JavaScriptu, místo toho aby byla renderována na serveru. Díky toho dosahuje kratší prodlev při navigaci po webu.

[9] Package manager pro NodeJS

[10] https://github.com/facebook/create-react-app

[11] https://github.com/mui-org/material-ui

6.2 Webová aplikace

Aplikace je rozdělena na tři stránky. Nejdříve uživatel narazí na login page. Obsahuje pouze tlačítko pro přihlášení, které využívá `firebase` knihovny a řídí celé flow autentizace. Přihlášení probíhá pomocí účtu Google. Další stránka obsahuje nastavení webhook klíče, který je potřebný pro aktivování triggeru na straně IFTTT. Hlavní stránkou je ale seznam webhooků, které lze přidávat, odebírat, editovat a spouštět. Pro jejich přidání a editaci je připraveno modální okno s nastavením vlastního názvu a jménem eventu, které musí korespondovat s tím na IFTTT. Layout stránek je rozdělen na tři části – horní lišta, postranní lišta a hlavní obsah. Pro navigaci je využita lišta na straně s odkazy na webhooky a nastavení. Horní lišta nabízí možnost odhlášení uživatele.

```
handleRunWebhook = ({ id, eventName }) => async () => {
  const { webhookKey } = this.state
  const url = `https://maker.ifttt.com/trigger/${eventName}/with/key/${
      webhookKey}`
  const settings = {
    method: 'POST',
    mode: 'no-cors'
  }

  try {
    await fetch(url, settings)
    await firebaseApp.functions().httpsCallable('runWebhook')({
      webhookId: id
    })
  } catch (error) {
    console.log(error)
  }
}
```

Výpis 19: Ukázka kódu pro spuštění webhooku

6.3 Firebase

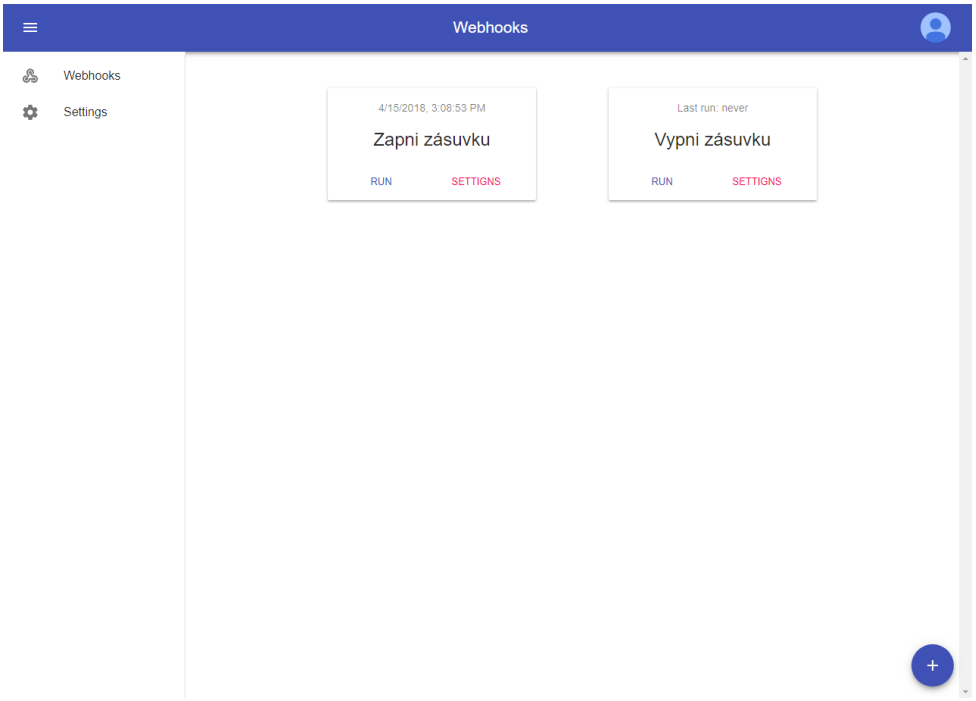
Většina konfigurace Firebase se provádí z webové konzole. Ta kromě nastavení nabízí i užitečné nástroje pro ladění, čtení logů nebo analýzu chování uživatelů. Část nastavení lze spravovat i lokálně z PC pomocí npm balíčku `firebase-tools`, který se používá jako CLI aplikace. Kromě nastavení slouží i jako lokální server a pro deploy aplikace do cloudu.

Pro správné fungování aplikace je ve správě autentizace povoleno přihlášení pouze pomocí Google účtu, jsou ale k dispozici i další jako Facebook, Twitter nebo kombinace emailu a hesla.

Dále je uživatelům povolen přístup do databáze Cloud Firestore pouze pro čtení. Všechny akce, které modifikují data, jsou totiž implementovány pomocí functions, které pracují v admin módu a mají přístup ke všemu. Pro fungování aplikace jsou potřeba čtyři functions. Jedna reaguje na event vytvoření nového uživatele a připraví pro něj dokument v databázi. Další tři slouží pro uložení, spuštění a smazání webhooku.

```
exports.createUser = functions.auth.user().onCreate(event => {
  db
    .collection('users')
    .doc(event.uid)
    .set({
      name: event.displayName,
      email: event.email,
      webhookKey: null,
      webhooks: []
    })
})
```

Výpis 20: Ukázka kódu pro vytvoření záznamu o uživateli po registraci



Obrázek 10: Výsledná webová aplikace

7 Závěr

V této práci jsem objasnil, co to je integrační platforma jako služba a její vzrůstající důležitost jak pro firmy tak i uživatele. Dále jsem tyto služby porovnal na příkladu z praxe, který ukázal, že ne všechny služby jsou si rovny. Zapier a především IFTTT se jeví jako služby, které je dobré zvolit pro základní problémy integrace. Naopak Microsoft Flow s Integromatem jsou komplexní nástroje, ve kterých je možno zpracovat i mnohem složitější toky a transformace dat.

V další části jsem zmapoval nároky na implementaci IFTTT protokolu, který je nutno implementovat v případě, že se firma rozhodne pro zveřejnění své služby. Bohužel se mi nepodařilo vytvořit a přidat vlastní službu přímo na IFTTT, jelikož pro navázání partnerství je třeba mít hotový produkt a jsou vyžadovány vysoké měsíční poplatky. I tak ale toto studium používaného rozhraní pro mě bylo přínosem. Dozvěděl jsem se totiž, jakým způsobem jsou navrhovány API, u kterých se od počátku počítá s tím, že musí mít dlouhou životnost, ale zároveň poskytnou jistou míru rozšiřitelnosti. A při tom všem budou využívány velkým počtem lidí, které tyto změny ovlivní. Musí se tedy dbát na to, aby byl co nejmenší počet tzv. „breaking changes“.

V případové studii jsem implementoval novou webovou aplikaci s využitím moderních technologií a serverless architektury. Je to architektura vhodná pro rychlý developement či prototypování. Výhodou může být využití pouze jednoho programovacího jazyka jak pro backend tak i fronted. Využití připravené `firebase` knihovny zase značně zjednodušuje využití OAuth autentizace oblíbených služeb třetích stran využívaných pro přihlašování k aplikacím.

Literatura

[1] What is iPaaS? Gartner Provides a Reference Model | MuleSoft [online]. *MuleSoft, Inc.* [cit. 2018-04-22]. Dostupné z: https://www.mulesoft.com/resources/cloudhub/what-is-ipaas-gartner-provides-reference-model

[2] Enterprise Integration Platform as a Service Software Reviews [online]. *Gartner Inc.* [cit. 2018-04-15]. Dostupné z: https://www.gartner.com/reviews/market/enterprise-integration-platform-as-a-service

[3] IBM - Modely cloudových služeb IaaS, PaaS, SaaS - Česká republika [online]. *IBM Česká republika, spol. s r.o.* [cit. 2018-04-15]. Dostupné z: https://www.ibm.com/cloud-computing/cz-cs/learn-more/iaas-paas-saas/

[4] See all services - IFTTT [online]. *IFTTT Inc.* [cit. 2018-04-15]. Dostupné z: https://ifttt.com/search/services

[5] Explore All Apps | Zapier [online]. *Zapier Inc.* [cit. 2018-04-15]. Dostupné z: https://zapier.com/apps

[6] Custom connector content links - Microsoft Flow | Microsoft Docs [online]. *Microsoft Corporation* [cit. 2018-04-15]. Dostupné z: https://docs.microsoft.com/en-us/flow/register-custom-api

[7] Apps & Services | Integromat [online]. *Integromat s.r.o.* [cit. 2018-04-15]. Dostupné z: https://www.integromat.com/en/integrations

[8] API requirements for building on IFTTT - IFTTT Platform [online]. *IFTTT Inc.* [cit. 2018-02-27]. Dostupné z: https://platform.ifttt.com/docs/api_reference

[9] Hardt, D., Ed. The OAuth 2.0 Authorization Framework [online]. *RFC Editor* [cit. 2018-02-27]. Dostupné z: https://tools.ietf.org/html/rfc6749

[10] Usage Statistics of Character Encodings for Websites, April 2018 [online]. *Q-Success* [cit. 2018-04-18]. Dostupné z: https://w3techs.com/technologies/overview/character_encoding/all

[11] Android Integrations | Zapier [online]. *Zapier Inc.* [cit. 2018-04-15]. Dostupné z: https://zapier.com/apps/android/integrations

[12] WICKRAMARACHCHI, Anuradha. Go Serverless with Firebase In: *Towards Data Science* [online]. 2018-09-02 [cit. 2018-04-22]. Dostupné z: https://towardsdatascience.com/go-serverless-with-firebase-5348dedb70e9

A Obsah přiloženého CD

Struktura složek a souborů na přiloženém CD.

- `/`
  - `web-application` ..... Zdrojový kód webové aplikace
    - `functions` ..... Zdrojový kód cloud functions
    - `public` ..... Statický obsah
    - `src` ..... Zdrojový kód SPA
    - `.firebaserc` ..... Soubor s nastavením Project ID
  - `thesis.pdf` ..... Soubor s touto prací

B Instalace a spuštění webové aplikace

Pro spuštění celé aplikace je zapotřebí NodeJS ve verzi 9.11.1 a Google účet k přihlášení se na Firebase. Pro kompletní instalaci lze postupovat v následujících krocích.

1. Instalace Firebase vývojářských nástrojů `npm install -g firebase-tools`
2. Přihlášení se ve vývojářských nástrojích účtem Google `firebase login`
3. Ve složce `functions` nainstalovat dependence `npm install`
4. Ve složce `web-application` nainstalovat dependence `npm install`
5. Ve složce `web-application` spustit build `npm run build`
6. Ve webové konzoli[12] vytvořit nový projekt s libovolným názvem a poznamenat si `Project ID`
7. V souboru `.firebaserc` nahradit `iftttwebhooks` za `Project ID` z minulého kroku
8. Developement server spustit příkazem `firebase serve`
9. Aplikace běží na adrese `localhost:5000`

[12] https://console.firebase.google.com/